$ 1.50

denver apple pi Seed

Newsletter for Denver's Apple users

Volume 7 Number 2 FEBRUARY 1985

Copyrights

by Jeff Beehler

Many new computer users ask about the different types of software. Many wonder why it is legal that you can copy from Apple Pi's software library but it is illegal to copy other types of software. This month I hope to explain the ins and outs of software copying.

Most computer software that you will find in a computer store, such as Apple Writer, is what we call commercially copyrighted software. This type of software is strictly ILLEGAL to copy and distribute among friends. This product protected by law so that only the author can authorize the copying of his software. It will cost you any where from $25.00 to $650.00 to purchase this software. I strongly suggest that you try it out before purchasing it. Many wonder how legal it is to copy software for backup purposes. Depending on the software, it may be okay to backup the disks for your own purposes, but not for other computer users. Always make sure you read what is called the "Licence Agreement". It will set down the 'rules' which you are supposed to follow to use that software legally. Read carefully.

Apple Pi's library consists of mainly of what we call PUBLIC DOMAIN software. This means that anyone can copy this software and many times the author does not request any type of compensation. Surprisingly enough, this software is sometimes better than it's commercial counterpart sold in computer stores. Before buying software at a computer store, check out our library and see what you can find. If there is something there similar to what the computer store sells, you may end up saving yourself some money.

There is another type of software that requests that you send either a donation or a registration fee to the author, if you have copied the software and are using it. We sell an example of this software entitled 'Diversa-DOS'. The author encourages you to send a set fee to his in order to use his software. Actually in many cases, it is against the law to copy and use this type of software without sending the author his required fee. One nice thing about this type of software is that you are free to examine it thoroughly and if you don't like it, you can erase it and you haven't lost anything.

Of course there are many subsets to each one of these types of software, but they all follow these guidelines that I have outlined above. If you yourself have written any programs that you are interested in contributing to our software library so that it may get spread around the area and maybe around the country, feel free to contact our librarian at any of the meetings or call the Apple Pi hotline. If you have any questions about the different types of software, feel free to contact me either at the meeting or via our hotline.

☆ contest ! ☆

The Seed is having a programming contest! What we want is Applesoft BASIC programs with no more than 5 lines of code. That's right only 5 lines. How much can you get out of only 5 lines? We will print the 1st, 2nd, and 3rd place winners in the Seed along with the programs. And as if this was not enough there will br prizes! 1st place will win a library storage case. 2nd place will win a box of 10 ESK diskettes and 3rd place will win a DOM. All the programs will be put on disk and made available to members. So what are you waiting for....show us what you can do with 5 lines. Contest ends March 10th. Bring your entry in on disk, over the modem on the BBS or mail in hard copy. You may chain as much to a line as you can but please no Binary code appended to the program.

☆ ☆

$2

denver apple pi

+:+:+:

Assemebly stuff

Assembly and DOS 3.3 Tips
by André van Meulebrouck

This article consists of tips for Assembly programming, especially in the DOS 3.3 environment.

First of all, why use Assembly if you are not a masochist? Assembly language is the most flexible language there is. It results in the most compact code, and offers the strongest I/O capabilities of any language.

Like alot of people at Apple Pi, I use the assembler called: ORCA/M. Curiously, ORCA/M spelled backwards is "macro", and indeed, it is a macro assembler.

Macros are text replacements for commonly desired high level language ideas and concepts. Macros are expanded to their equivalent in Assembly code, and they then replace the macro statement that requested the expansion. This means you can use some of your favorite statements from higher level languages without having any understanding of how they would be coded in Assembly. This makes Assembly language so much easier that it becomes like a high level language, and a very attractive and viable alternative to Basic.

Any DOS commands you can do in a running Applesoft program by preceeding a print statement with a control d can also be done in Assembly programs in essentially the same manner.

Many people do not call DOS this way when accessing files on disk, and instead, make direct calls to the low level routines of DOS that deal with the disk. Normally, when the DOS command interpreter gets a request to do a disk access, it calls the FM (File Manager) after setting up some parameters, then, the FM call RWTS (Read/Write a Track or Sector) after some parameters are set up. You really have to know what you are doing to make direct calls to RWTS, and when you write to disk via a direct call to RWTS, the FM will not know what you have done, because RWTS is a lower level routine than the FM. I think it is good to let the FM know what you are doing so it can update the VTOC (Volume and Table of Contents). In other words, you can write to disk without reference to file names and what already resides on disk by calling RWTS directly, but if you later call DOS or the FM, what you wrote to disk by calls to RWTS may get written over because the FM will not be aware there was anything there because its "records" (VTOC) will not have been updated. Some people instead of calling RWTS directly, call the FM directly. There are alot of details and parameters that DOS takes care of when it calls the FM, and the FM takes care of alot of details before calling RWTS. If you call either directly, you have to take care of all those machine specific details yourself. If you wish to access the disk the way DOS does, and you aren't going to do anything different from the way DOS does it, you can call DOS to do your file handling in your assembly code just like you do it in Basic. There are some time savings from calling RWTS or the FM directly, but it is more effort, and more work is involved to assure reliability. Also, machine specific details that don't directly contribute to your algorithm may take time away from your planning of the overall picture.

Output to the screen is done using the COUT routine (character out), and DOS interrogates each character that is sent to this routine. If it gets a control d, the following characters it receives from calls to COUT until a carriage return is recieved, will be interepreted as a DOS command. Before you do this, do a call to CROUT (carriage return out) to clear out the DOS command buffer, in case there is anything already in it. Then, load the accumulator with a control d. A control d is $04 on the ASCII chart, but you must remember that the Apple II series uses the highest bit (bit 7) of characters to indicate inverse, and normal modes. (Bit 6 is used to control flashing). In normal mode, the high bit will be set. That means a control d is $84. After the control d, use repeated calls to COUT to send your command to DOS, character by character. At the end of the command, do a call to CROUT to finish the command. The address of COUT is $FDED. The address of CROUT is $FD8E. The use of the ORCA/M assembler makes this even easier with its print macro. Example:

```
        KEEP 'SAMPLE.PROGRAM',V254
*  SAVE OUTPUT TO DISK INITIALIZED AS VOL. 254
        MCOPY MACLIB.I/O,V103
*  GET THE MACRO LIBRARY THAT HAS PRINT MACRO
BEGIN  START
*  TELL ASSEMBLER TO START ASSEMBLING
COUT   EQU   $FDED   ! ADDRESS OF ROM ROUTINE TO
*                      OUTPUT A CHAR TO SCREEN
```

```
CROUT  EQU   $FD8E    ! ADDRESS OF ROM ROUTINE TO
*                      OUTPUT A CARRIAGE RETURN
       JSR   CROUT    ! CLEAR OUT DOS COMMAND BUFFER
       LDA   #$84     ! LOAD ACCUMULATOR WITH CONTROL
*                      D FOR DOS COMMAND
       JSR   COUT     ! OUTPUT CONTROL D
       PRINT 'CATALOG D2' ! OUTPUT ASCII CHARS OF

*                           DOS'S CATALOG COMMAND
       END            ! TELL THE ASSEMBLER TO STOP
*                      ASSEMBLING
```

The print macro in the above example will be expanded by the ORCA/M assembler to a loop that loads each character of the DOS command, and outputs them by calling COUT after each one is loaded. If your Assembler does not have such a macro, you would do it like this:

```
       KEEP 'SAMPLE.PROGRAM',V254
BEGIN  START          ! START ASSEMBLING

* DEFINE CONSTANTS

COUT   EQU   $FDED    ! ADDRESS OF ROM ROUTINE TO
*                      OUTPUT A CHAR TO SCREEN
CROUT  EQU   $FD8E    ! ADDRESS OF ROM ROUTINE TO
*                      OUTPUT A CARRIAGE RETURN

       JMP   PROG     ! JUMP OVER DATA/VARIABLE SPACE

* ESTABLISH DATA AND VARIABLE SPACE

CATDAT DC    H'84'    ! HEX DATA FOR CONTROL D WITH
*                      HIGHEST BIT SET
       DC    C'CATALOG'  ! DATA FOR CATALOG COMMAND
       DC    H'0'     ! ZERO THE LOOP WILL LOOK FOR
*                      TO KNOW WHEN IT'S DONE

PROG   JSR   CROUT    ! CLEAR OUT DOS COMMAND BUFFER
       LDX   #$00     ! CLEAR OUT X REGISTER

LOOP   LDA   CATDAT,X   ! LOAD ACCUMULATOR WITH ONE
*                        CHAR OF DOS COMMAND
       BEQ   DONE     ! IF IT IS ZERO, THE ENTIRE
*                      COMMAND HAS BEEN SENT OUT
       INX            ! INCREASE OFFSET TO LOOK AT
*                      NEXT CHAR OF DOS COMMAND
       JMP   LOOP     ! DO NEXT CHAR

DONE   JSR   CROUT    ! FINISH COMMAND AND SEND TO
*                     DOS BY DOING A CARRIAGE RETURN
       END            ! STOP ASSEMBLING
```

A note on style here. I like to put all my constants first in alphabetical order, then all variables and data. Although the above example could have been written in less code with more efficiency, I place a higher emphasis on structure than efficiency. This is because my highest priority is assuring reliablity. Reliability is easier to verify if there is some structure in your program, especially when you are writing long programs. I have however, overdone the comments and structure for the benefit of people that are new to Assembly.

If you run the above programs, and your drive is not working properly, the program would crash when it attempts to access the disk. By crashing, I mean that your program would be aborted, and control would be returned to the immediate mode of DOS/BASIC. This might make your program appear to be user-hostile, and would make reverse engineering attempts easier, because you could then drop into the monitor and look around.

If you want your program to maintain control, even in the event of a DOS error, you can. Errors may consist of a forgetful user leaving a drive door open, or deliberate abuse of your program.

The first tip I will relate allows you to have control returned to your program at any point you desire after a DOS error occurs. This technique makes use of the DOS error re-entry vector that determines where control is returned to after a DOS error. A vector is an address that holds the address of something of interest. This is an example of indirection. This is done because the ultimate address of interest may change, but the address we go to, to find out where the ultimate address is, is always known since IT does not change. The DOS vector at $9D5A and $9D5B holds the address of the immediate mode of DOS/BASIC as a default. In a running program, if an ONERR statement is used, the vector at $9D5A will have the address of the basic program line that you specified in your ONERR statement. In your Assembly language programs, you can put the address of one of your own routines at this vector. Presumeably, you would put the address of an error routine you have written to handle DOS errors as you would like to handle them. This technique only works if you originate your assembly language program from a Basic program (i.e. 10 PRINT CHR$(4);"BRUN FOO").

President notes

I am sorry to report that after the January Core Meet, the Apple PI board of directors found it necessary to rescind the membership of one of our members for flagrant disregard for the United States Copyright laws. This action as taken after much discussion on the issue.

Apple Pi will only support the sale of original software with its original documentation at club sanctioned activities unless an agreement with the author allows such sales, i.e. Diversi-DOS. This policy also includes the photocopys of software documentation.

Apple Pi experimented with a different projection system at the January meeting using an Electahome projection unit rented from an audio visual company. This unit was required in order to have a projection unit capable of the wide bandwidth requirements necessary for Macintosh projections. The modified Macintosh was provided through the efforts of the Denver Apple Sales Office. Our thanks to Apple Computer for their continued support to the club. As most members present will attest, the unit did an excellent job on the Apple II computer in 40 column and fair on 80 column. Presentation of the Macintosh screen was only fair in the front center of the room and poor in the back and sides. Unfortunately, it is the only unit we are aware of that will project the Macintosh screen. We shall look around to see if we can find such a unit on a regular basis, at a reasonable price to replace the CSM Advent which we have been using over the past years. If you know of where one of these units can be obtained/borrowed reasonably, please contact me directly.

Remember the Core Meet in March will include a Garage Sale for the membership. Also, looking into the future, there will be no meeting on July 4th. I want to start telling people now so they can mark their calendars.

KIDS

APPLESEED NOTES

Interested in an inexpensive alphabet program designed for ages 2 through 5? Apple Pi Appleseeds has such a program to offer. "ELFWARE" is sponsored by International Apple Core. It not only has very appealing graphics, but the accompanying music is fascinating to listen to. Early childhood alphabet software should always incluce lower case letters, as well as capital letters. This program has both - the capital letter is featured with the picture, and the word description of the picture is printed in lower care - many children gain sight reading of the word along with learning the individual letter! Once booted, even the youngest child can handle the program by themselves. It also teaches some keyboard familiarity. A color t.v or monitor is very presferable, but not entirely necessary. ELFWARE will be available at the next CORE meet, the cost is $5.00 initially, then if you are satisfied with the program there is an address on disk to send an additional $17.50 - a total of $22.50, not bad compared to the average price of $30.00 to $40.00 per program for quality early childhood software. If you have any questions about this program, or other children's software please feel free to call Liane Nusse, 279-5763.

APPLESEEDS will demonstrate various childrens programs at the Feb. Core Meet,
hope to see you there!

FREE 30 DAY TRIAL

OF

Snowfile

Snowfile is an easy-to-use database program that is designed to handle all of your filing needs. Snowfile can store a phone list, a mailing list, an inventory list, or any other file you could keep on paper in a conventional filing cabinet. Best of all, you don't have to know a lot about your computer to operate Snowfile. One Snowfile user said "I learned enough to get me going in it within 1 minute without reading much of the directions."

In the software market today, it's hard to find good software for a reasonable price. That's why we think it's a good idea to have a 30 day free trial option on software.

Here's how it works: all you have to do is complete the coupon below and send it to us. We'll mail you a trial copy of Snowfile right away. Try it for 30 days. If you decide Snowfile isn't right for you, just send it back to us at the end of 30 days and you owe nothing. If you decide to keep it, just send us a check for $29.95. If we haven't received a response from you within 45 days, we'll assume you wish to keep Snowfile and we'll invoice you for $29.95. Sound easy? It is.

Snowfile works on any Apple //c, //e, or 64K][+. Send in your coupon today.

___ Yes, I'd like to give Snowfile a try. Please send me a copy to try for 30 days. At the end of 30 days, I will return the copy or send you a check for $29.95.

I have an Apple: (__) //c (__) //e (__) //+

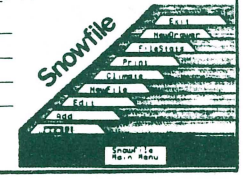
Name: _____

Address: _____

City: _____ State: _____

Zip: _____ Home phone: _____

MAIL TO: SNOWBOUND TECHNOLOGIES
11735 WEST 72ND PLACE
ARVADA, CO 80005



Tres. $

INCOME		
New Members		1036.00
Renewal Members		450.00
DOM Sales		335.00
Library Copies		349.00
Sales of Supplies		1815.00
Sale of Equipment		130.00
Interest Earned		0.00
TOTAL INCOME		4115.00
EXPENSE		
Supplies for Resale		3681.67
Hot Line		0.00
CSM Room Rent		300.00
Printing Costs		174.82
Postage		68.16
Supplies & Copies		25.00
BBS Expense		0.00
State Sales Tax		0.00
Christmas Party		45.98
Equipment Purchase		1390.63
TOTAL EXPENSE		5686.26
ENDING CASH BALANCE	12-31-84	4509.89
ASSETS	12-31-84	
Cash on Hand		25.00
Cash in Bank		4509.89
Inventory for Resale		5170.27
Prepaid Expense		381.03
Accounts Receivable		0.00
Equipment		0.00
Total Assets		10086.19
LIABILITIES		
Sales Tax Accrued		260.71
Accounts Payable		0.00
Unposted Income		0.00
Total Liabilities		260.71
NET WORTH		9825.48

THANK YOU...

When you stop in to an Apple Pi member store tell them that you are with Apple Pi and thank them for the support they give. Our store members make valuable contributions to Apple Pi from special discounts to the door prizes at the meetings. When you need hardware, software or computer supplies remember our store members.

adam's computers, inc.

3355 So. Wadsworth Blvd.
Lakewood, CO 80227 (303) 986-9655

colorado Computer Systems, Inc.

3005 West 74th Avenue
Westminster, CO 80030 (303) 426-5880

ComputerLand®

8749 Wadsworth Blvd.
Arvada, CO 80003 (303) 420-1877

ComputerLand®

2422 So. Colorado Blvd.
Denver, CO 80222 (303) 759-9140

The Neighborhood COMPUTER STORE

13045 West Alameda Parkway
Lakewood, CO 80228 (303) 988-9140

JOHNNY APPLESEED

Dear John,

I am a new owner of an Apple //c. When I bought my Apple //c, it came with a disk labeled "PRODOS" and a disk labeled "DOS." I have been using the "PRODOS" disk to write small BASIC programs with. I had been doing pretty well until I purchased your January Disk-of-the-Month. I have decided that it must use "DOS" because I was having trouble working it. Can you give me some quick tips to using DOS?

PRODOS OR NODOS

Dear NoDOS,

Your question is a very good one. Yes, our Disk-of-the-Months use a modified DOS format. This is because 1) Apple Pi doesn't have permission from Apple to distribute PRODOS on our DOMS (we may distribute ProDOS only if the disk is from an official ProDOS distributer) and 2) most of the programs that we put on the DOMS are originally based in DOS.

If you would like, you can convert most of the programs on our DOS disk of the months to PRODOS using the convert utility on your ProDos users disk. While that is possible, it would probably be easier to just learn a few things about dos. Here are some tips:

When you first boot one of our DOMs (put it in to the disk and turn the computer on), usually there will be a title page and then you will see the "]" prompt. At that time, type "CATALOG". You will then see a list of files on the disk. Here is an example:

```
*A 005 GAME
*B 008 MENU
*I 014 GRAPHICS
*T 002 INFO
```

This example contains the four types of file you will see most, A, B, I, and T. A stands for Applesoft. To run the file titled "GAME", just type RUN GAME. This will work for any Applesoft (A) file.

B stands for binary file. To run the file titled "MENU", type BRUN MENU. This will work for most binary files.

I stands for integer basic. To run the file titled "GRAPHICS", type RUN GRAPHICS. IF you get the error "LANGUAGE NOT AVAILABLE", then boot your DOS system master disk, wait until you see the "]" prompt, insert the DOM, and type RUN GRAPHICS again.

T stands for text. Do not attempt to run a file that is a text file. It will be used by one of the other programs on the disk.

Here's a review:

File type	What to do
A	RUN xxxxx
B	BRUN xxxxx
I	RUN XXXXX (see text)
T	nothing

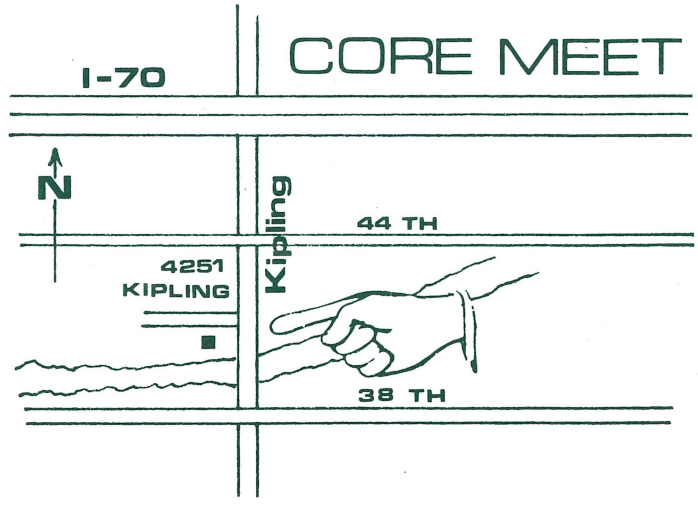
There are two other file types that you may have seen on other disks (not DOSs). These are "S" and "R" files. You will also see these in ProDOS. The "S" stands for System file and in DOS is not usable by the human (only other software). The "R" stands for relocatable code and is sometimes used by more sophisticated programs.

I hope that these quick tips help.

John

Questions? Send them to:

Johnny Appleseed
% Denver Apple Pi
P.O. Box 17467
Denver, Colorado 80217-0467



If you do not want to originate your Assembly program in BASIC, there are three addresses that can be manipulated to fake out DOS into thinking you are in BASIC. It is often desireable to be able to look like an Applesoft program to DOS when you are relating to DOS. This is because DOS was especially designed to support Applesoft, so Applesoft is, in many ways, DOS's language of choice, with all the special treatment that implies. Here are the three locations and what to put in them;

1. Prompt symbol at $33. Put a $06 here. In immediate mode, there is a $DD here (the "]" character). In a running program, DOS expects a $06 here.
2. Language being run at $AAB6. Put a $40 here. DOS will then think FP BASIC is running.
3. Line number of program being run at $75 and $76. DOS will think a program is running if there is a non-$FF value in the high byte ($76). If the value at $76 is $FF, DOS thinks you are in immediate mode.

If you set the above three locations, an added advantage is that you can access text files. When an Assembly program is running, DOS normally thinks it is in immediate mode. In immediate mode, DOS will not let you access text files.

When a DOS error occurs, DOS will beep and display an error message. If you wish to suppress the beep and DOS's error message, set the high bit of location $D8. It is possible to set one bit of a location without disrupting the other bits at the location. This is done by creating a "mask" and doing a logical operation with the mask and the data location. The mask is set-up to deliberately change only the bits you wish to change. In an OR operation, if your mask supplies a 1 for a given bit position, that guarantees a 1 in the result at that position. If you supply a 0 in your mask, the result after an OR will simply be whatever the data bit was at that position. So, if your mask is 1000 0000 in base two, and you do a logical OR operation, the result will be the same as the data used, except that the high bit will be set. In 6502 mnemonics, this would be:

```
ERSPRS EQU $D8     ! ERSPRS = ERROR SUPPRESS LOCATION
       LDA #$80    ! LOAD MASK, 80 HEX = 1000 0000 IN
*                    BASE TWO (BINARY)
       ORA ERSPRS  ! LOGICAL OR MASK WITH ERROR
*                    SUPPRESS LOCATION
       STA ERSPRS  ! PUT RESULT BACK IN ERROR
*                    SUPPRESS LOCATION
```

In order to find out what error occured after a DOS error, there are two things you must do. First, you should save the X register. DOS puts the error number of the error that occured in the X register before it returns control to the address stored at the $9D5A vector. This value is stored at $DE when a BASIC program is running, if the program had an ONERR statement in it. Since BASIC uses $DE for this purpose, that is a logical place to put it when using Assembly language. The error messages that correspond to the error numbers used by DOS are listed below. These numbers and error messages are the same ones you would check for in BASIC after an ONERR statement because you ARE in BASIC as far as DOS is concerned.

1	LANGUAGE NOT AVAILABLE
2,3	RANGE ERROR
4	WRITE PROTECTED
5	END OF DATA
6	FILE NOT FOUND
7	VOLUME MISMATCH
8	I/O ERROR
9	DISK FULL
10	FILE LOCKED
11	SYNTAX ERROR
12	NO BUFFERS AVAILABLE
13	FILE TYPE MISMATCH
14	PROGRAM TOO LARGE
15	NOT DIRECT COMMAND

So, with the tips mentioned so far, you can see that programming in Assembly does not need to make life difficult, and, it should be easy to write your entire program in Assembly. That saves you from having to interface a Basic program and an Assembly one, in other words, you won't have to worry about setting up data/variable communication between a BASIC program and an Assembly one. This makes your program clean and "simple".

Since you now don't have to originate in BASIC, you can make your Assembly program self-booting by initializing a disk using the name of your Assembly program as the hello program. To do this, you will have to change a location in memory first. There are two ways to do this. You can get into the monitor (type "CALL -151" without the quotes), then type "9E42:34" (without the quotes). That will put a hexidecimal 34 into address $9E42. The other way you can do this is by converting the address and number

from hexidecimal into decimal and do a poke. Not originating in BASIC might be desireable to make reverse engineering of your program more difficult.

One last security concern you may have is to stop your program from crashing in the event of a reset (hitting control and reset at the same time). Doing a reset activates the reset vector which is located at $3F2 and $3F3. If you change the high order byte ($3F3) you can get the Apple to do a reboot on resets. Putting a 0 into $3F3 will accomplish this. After each reboot, the reset vector must be changed again since a reboot reinitializes this vector. Rebooting on resets is a simple and popularly used method of dealing with the reset vector in commercial programs.

For the benefit of people with older Apples that don't have reboot capability (other than turning the machine off and on), you might give a boot option in your program. There is a ROM routine which searches the slots of the Apple for a drive controller card, and then transfers control to the firmware on it to do a reboot. Doing a call to $FAA6 will activate this. Since you will never return to your program after such a call, don't use a JSR, use a JMP instead. Example:

```
JMP $FAA6  ! REBOOT
```

In Assembly language, you are at a very low level, language-wise. This has a tendency to distract your thoughts away from algorithms to machine specific details. Try not to get too caught up in bit twiddling for a few clock cycles of extra efficiency or you may wind up having unreliable code, losing sight of the overall algorithm, and/or taking "forever" to finish your program.

Finally, good luck!

CLASSIFIED AD

Are you a "computer intimidated" spouse? Would you like to learn how much your computer could do for you as a homemaker, volunteer, Avon Lady, etc.? Would you like to know how to get your kids interested in computers and how to purchase the right software to teach them?

Call Nusse Computer Services, located in Golden, for "beginner's beginning" tutoring with understanding guidence. $10.00 per hour - 279-5763.

We Need You !!

Are you sitting there, reading this, wondering what you can do to be a part of Apple Pi??? Here's your chance: we need new programs for our library, LOTS OF THEM!!! If you have any programming ability at all, sit down and write a program that we could include on our Disk-of-the-Month or another library disk. It doesn't have to be long or fancy, just written by a person who cares about Apple Pi.

We are a 100% volunteer club, and our library is made up of volunteer contributions. We don't pay with money for the programs you contribute to the library, we pay with pride. Right now, the Apple Pi library is something any member can be proud of...but we have to keep it that way. We can't do this without your help...so contribute something...anything to keep our library going strong.

Some of you probably know that I ask for contributions almost every month, but I have a good reason. Apple Pi as a club is growing rapidly in membership. Our library is not. Every month for the last 4 or 5 we have been getting less and less contributions...until last month it came to zero. We can't force you to contribute to our library, so you have to force yourself. Take some time today to write a short program.

I would also like to encourage you to complete and return the membership directory questionnaire located elsewhere in this issue. At the bottom of this questionnaire are the three most important questions of all: will you give a small amount of your time to Apple Pi? I urge you to check "YES" on at least one of the last three questions.

--> Trent Hein, Librarian <--

denver apple pi

Green Apples

Have you ever wanted a way to enter a date that would convert the input of numbers into a standard month/day/year format. This little program will let you take this: 3/13/85 and will give you back this: Mar. 13, 1985. The routine is intelligent enough to know that there are not more that 12 months nor are there ever more than 31 days in a month. The routine will not check to see if you are putting 30 days in February but you can add this if you want. You will also find some little sound routines to make things interesting.

The program is designed for an Apple computer using an 80 column card. If you wish to change this to 40 column you will need to change the print statements and the tabs (POKE 1403 is an 80 column HTAB). You will also want to change the lower case strings into upper case if you are using an older Apple.

The only other note I will make is the way the month string works. All the possible months are in the string MO$. Each month is 3 characters long. Line 835 will take the number of the month * 3 (length of each months string) and then subtract 2 to fetch the first character. The string starts out "JanFebMar..." so if the month is 2 then 2*3-2= 4 so the computer will fetch three characters starting at position 4 (Feb). If the month was 3 then 3*3-2= 7 so the computer will fetch three characters starting at position 7 (Mar).

Have fun with this. If you have a program that you would like to share please send it in.

```
10  REM ##############################
15  REM #                            #
20  REM #   "GET DATE"  SUBROUTINE   #
25  REM #                            #
30  REM #     by  RANDALL ANSLEY     #
35  REM #                            #
40  REM ##############################
45  REM #     COPYRIGHT (C) 1985     #
50  REM ##############################
55  REM #                            #
60  REM #   THIS CODE MAY BE USED    #
65  REM #     IN ANY PROGRAM AS      #
70  REM #     LONG AS CREDIT IS      #
75  REM #     GIVEN TO R.ANSLEY      #
80  REM #    AND DENVER APPLE PI     #
85  REM #                            #
90  REM ##############################
95  REM
```

```
800 MO$ = "JanFebMarAprMayJunJulAugSepOctNovDec":DT$ =
    ". "
802 TEXT : HOME : VTAB 1: HTAB 1: FOR I = 1 TO 80: PRINT
    "=";: NEXT I
804 VTAB 3: HTAB 1: FOR I = 1 TO 80: PRINT "=";: NEXT
    I
806 VTAB 2: HTAB 24: PRINT "--------------DATE------
    ---------"
808 POKE 34,5
810 IF D1$ = "" THEN  VTAB 6: POKE 1403,29: PRINT "Th
    e Date is not loaded."
811 IF DI THEN  VTAB 6: POKE 1403,40 -  LEN (D1$) / 2
    : PRINT D1$
813 POKE 34,8
814 HOME :BZ = 20:DT$ = ". ": GOSUB 20: REM  BZ = LEN
    GTH OF BUZZ
815 VTAB 10: POKE 1403,27: PRINT "What is the MONTH
    (1 - 12)": VTAB 14: POKE 1403,33: INPUT "";D2
816 IF D2 < 1 OR D2 > 12 THEN  PRINT  CHR$ (7): GOTO
    814
817 IF D2 = 5 THEN DT$ = "  "
819 HOME : GOSUB 19: VTAB 14: POKE 1403,33: PRINT D2;
    " / "
820 VTAB 10: POKE 1403,27: PRINT "What is the DAY
    (1 - 31)": VTAB 14: POKE 1403,36 +  LEN ( STR$ (D
    2)): INPUT "";D3
821 IF D3 < 1 OR D3 > 31 THEN  PRINT  CHR$ (7): GOTO
    819
824 HOME : GOSUB 19: VTAB 14: POKE 1403,33: PRINT D2;
    " / ";D3;" / 198"
825 VTAB 10: POKE 1403,32: PRINT "What is the YEAR": VTAB
    14: POKE 1403,42 +  LEN ( STR$ (D2)) +  LEN ( STR$
    (D3)): INPUT "";D4
830 IF D4 < 0 OR D4 > 9 THEN  PRINT  CHR$ (7): GOTO 8
    25
835 D2 = D2 * 3 - 2:D2$ =  MID$ (MO$,D2,3):D3$ =  STR$
    (D3) + ", ":D4$ = "198" +  STR$ (D4)
840 D1$ = D2$ + DT$ + D3$ + D4$
850 POKE 34,5: HOME : VTAB 6: POKE 1403,34: PRINT "Th
    e Date is:"
855 VTAB 8: POKE 1403,40 -  LEN (D1$) / 2: PRINT D1$
857 POKE 34,8
860 VTAB 20: POKE 1403,25: PRINT "Is the date correct
    ?  (Y/N)  ";: GET Z$
865 IF Z$ = "Y" OR Z$ = "y" THEN DI = 1: RETURN
870 IF Z$ = "N" OR Z$ = "n" THEN 814
875 PRINT  CHR$ (7): GOTO 860
880 FOR T = 1 TO BZ:SD =  PEEK (49200): NEXT T: RETURN
    : REM     KEY CLICKER
890 FOR TT = 1 TO 2: FOR T = 1 TO BZ:SD =  PEEK (4920
    0): NEXT T: FOR T = 1 TO 50: NEXT T: NEXT TT: RETURN
    : REM     LONG BUZZ
```

APPLE PI MEMBERSHIP DIRECTORY QUESTIONNAIRE

Please complete this form and mail it to Apple Pi, P.O. Box 17467, Denver, CO 80217 or hand it to any officer at any meeting. Initial deadline is 15-Mar-85.

(REQUIRED) Name: _____

(REQUIRED) City: _____

(Optional) Street Address: _____

(Optional) Zip code: _____ Home telephone: ____-____-_____

PLEASE CHECK THE CATEGORIES YOU ARE INTERESTED IN ONCE. PLEASE CHECK THE CATEGORIES YOU ARE WILLING TO HELP OTHERS IN TWICE.

A. _____ BUSINESS
B. _____ DATA MANAGEMENT/STORAGE
C. _____ EDUCATION
D. _____ GAMES
E. _____ MATHEMATICS
F. _____ CP/M
G. _____ BASIC
H. _____ FORTRAN
I. _____ MACHINE LANGUAGE
J. _____ PASCAL
K. _____ TELECOMMUNICATIONS
L. _____ UTILITIES
M. _____ WORD PROCESSING
N. _____ MacINTOSH
O. _____ LISA
P. _____ APPLE ///
Q. _____ COMPUTER ART/GRAPHICS DESIGN
R. _____ ENGINEERING
S. _____ INVESTING/TAXES
T. _____ PRODOS

Would you be willing to serve on a nominating committee for next years officers??:

Would you be willing to run for an office in the future???:

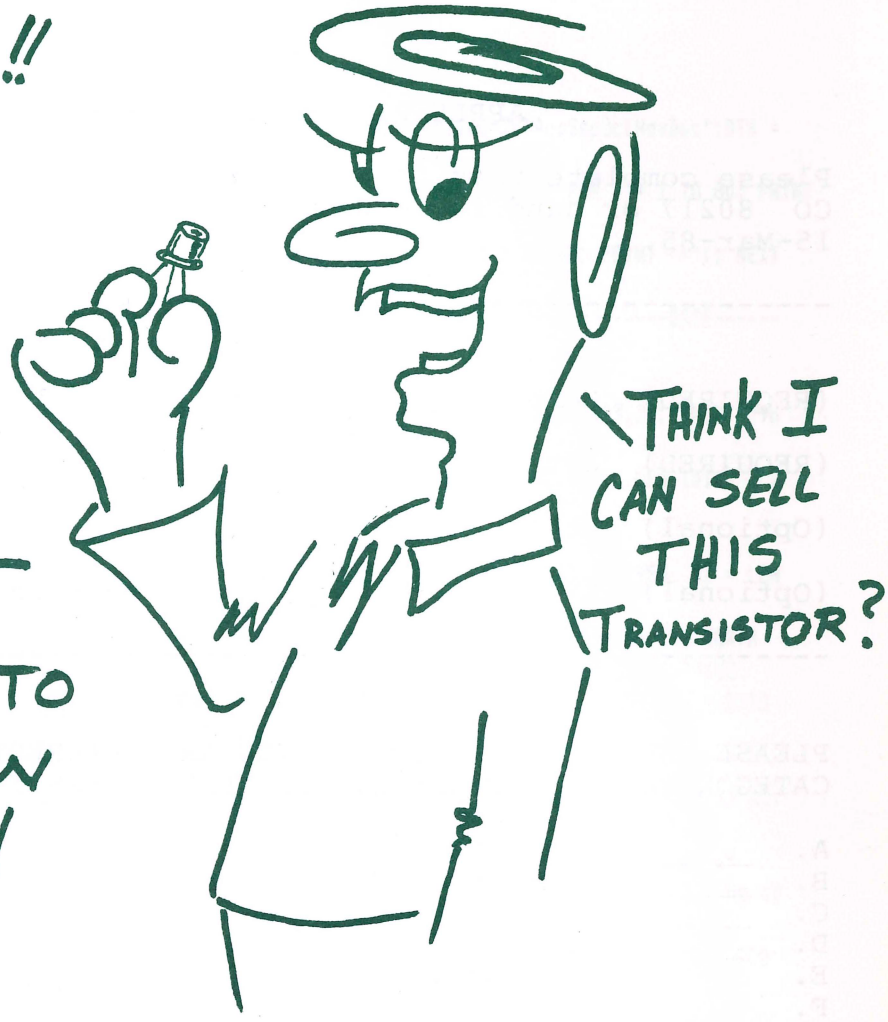
Would you be willing to assist one of the present officers???:

GET YOUR GOODIES TOGETHER!!

GARAGE SALE

AT THE MARCH 10TH CORE MEETING

IF YOU WANT TO SELL IT THEN BRING IT IN!!



---

denver apple pi
Seed

P.O. Box 17467
Denver, CO 80217

BULK RATE
U.S. POSTAGE
PAID
DENVER, COLO.
PERMIT NO. 3182

Price $1.50